Contents

Документация по обработке данных о профессиях

Магомедов Курбан

15 мая 2025 г.

1 Вступление

Данный проект разработан для преобразования неструктурированных текстовых данных о профессиях в структурированные числовые признаки, пригодные для использования в моделях машинного обучения.
Основные задачи решения:

- Очистка и стандартизация текстовых данных
- Коррекция орфографических ошибок
- Извлечение профессиональных характеристик
- Интеграция полученных признаков в исходный датасет

Решение построено на комбинации методов:

- Механической обработки текста
- Использования внешних Yandex Speller API для корректировки орфографических ошибок
- Использования DeepSeek API для получения характеристик профессий
- Пакетной обработки с сохранением промежуточных результатов

2 Требования

2.1 Программные требования

Python >= 3.9 и библиотеки:

- pandas
- requests
- openai
- tqdm
- json

Их можно будет установить с помощью pip install -r requirements.txt. Файл requirements.txt прилагается.

2.2 Требования и рекомендации API

- Yandex Speller API:
  - Лимит: 10,000 запросов в день
  - Рекомендуемая частота: не более 1 запроса в 0.3 секунды
- DeepSeek API:
  - Требуется действительный API-ключ (со средствами на балансе)
  - Рекомендуется запускать алгоритм в соответствии с расписанием скидок на обработку API запросов (см. сайт платформы deepseek)

3 Описание работы программы

3.1 Исходные данные

Исходные данные в нашей задаче это столбец с записями профессий *'occupation'*.

3.2 Предобработка данных

Предобработка состоит из двух этапов:

1. **Механическая очистка**.
   Используем функцию *clean*(). Её задача убрать все не алфавитные символы и лишние пробелы.
2. **Исправление орфографических ошибок**.
   Для этого мы используем функцию *yandexspeller*(). В этой функции генерируются батчи названий профессий для исправления и отправляются Yandex API.

3.3 Обработка данных с помощью DeepSeek API

Обработка данных представлена в функции *deepseekAPI*(). При запуске она будет отображать процесс обработки датасета. Этапы работы этой функции представлены ниже.

3.3.1 Формирование батчей

Чтобы сэкономить токены и время предложено объединять записи профессий в группы по 20-40 названий (батчи).

3.3.2 Промпт обработки

Каждый батч обрабаывается в соответствии с этим промптом:

```
prompt = f"""
Анализ профессий: {', '.join(group)}
Верни JSON, где ключи - названия профессий, а значения - словари с:
- "qualification_level" (1-10)
- "hierarchy_level" (1-10)
- "is_industrial" (0/1)
- "is_healthcare" (0/1)
- "is_management" (0/1)
- "is_security" (0/1)

Если параметр неопределим - ставь минимальное значение.
Если это не профессия - все параметры -1.
Только JSON без комментариев!
"""
```

где *group* это батч профессий. Причём температура модели выставлена низкая (0.3) так как эта задача не требует большого количества слов. Пример результата обработки батча:

```
{
профессия1: {свойство1: значение1, свойство2: значение2, ..., свойствоK: значениеK},
профессия2: {свойство1: значение1, свойство2: значение2, ..., свойствоK: значениеK},
...
профессияB: {свойство1: значение1, свойство2: значение2, ..., свойствоK: значениеK},
}
```

3.3.3 Алгоритм обработки

Для каждого батча вызывается вложенная функция *process_group*(). Каждый результат вызова этой функции сохраняется в промежуточном файле, который имеет такой формат:

```
{
    'профессия1, профессия2, ..., профессияB': {
    профессия1: {свойство1: значение1, свойство2: значение2, ..., свойствоK: значениеK},
    ...
    },
    'профессияB+1, профессияB+2, ..., профессия2B': {
    профессияB+1: {свойство1: значение1, свойство2: значение2, ..., свойствоK: значениеK},
    ...
    }
}
```

смысл такой обработки в том, чтобы исключения обрабатывались корректно и чтобы имелась исходная версия каждого результата обработки.

3.3.4 Сохранение результатов обработки

В конце из всех результатов обработок профессий будет сгенерирован результативный файл в котором будут перечислены все профессии с их свойствами

```
{
    профессия1: {свойство1: значение1, свойство2: значение2, ..., свойствоK: значениеK},
    профессия2: {свойство1: значение1, свойство2: значение2, ..., свойствоK: значениеK},
    ...
}
```

3.4 Интеграция результатов обработки с исходным датасетом

Сперва нужно загрузить словарь свойств профессий из предудыщего этапа

3.4.1 Подключение нового столбца

С помощью функции *turn_into_features*() генерируются новые столбцы свойств каждой профессии в исходном датасете.

3.4.2 Подсчёт качества обработки

С помощью функции *check_impute*() вычисляется качество обработки: процентаж пропущенных значений и значений, в которые языковая модель посчитала не относящимися к профессиям.

Список литературы